

WEST

## Freeform Search

Database:

US Patents Full-Text Database  
 US Pre-Grant Publication Full-Text Database  
 JPO Abstracts Database  
 EPO Abstracts Database  
 Derwent World Patents Index  
 IBM Technical Disclosure Bulletins

Term:

Display:  Documents in Display Format:  Starting with Number Generate: ☐ Hit List ☒ Hit Count ☐ Side by Side ☐ Image

Search

Clear

Help

Logout

Interrupt

Main Menu

Show S Numbers

Edit S Numbers

Preferences

Cases

## Search History

DATE: Monday, November 03, 2003 [Printable Copy](#) [Create Case](#)

Set Name Query  
 side by side

Hit Count Set Name  
 result set

DB=USPT; PLUR=YES; OP=ADJ

|           |  |       |           |
|-----------|--|-------|-----------|
| <u>L7</u> | L6 and (tag with cache with (modif\$8 or updat\$4))  | 60    | <u>L7</u> |
| <u>L6</u> | L5 same l4   | 212   | <u>L6</u> |
| <u>L5</u> | cache with ((store adj2 in) or modif\$8 or updat\$4)   | 6688  | <u>L5</u> |
| <u>L4</u> | buffer near8 (flush\$3 or clear\$3 or clean\$4 or empt\$4 or remov\$4 or eliminat\$3 or replac\$6) | 40110 | <u>L4</u> |
| <u>L3</u> | L2 and l1  | 4     | <u>L3</u> |
| <u>L2</u> | cache near8 (store adj2 in)  | 40    | <u>L2</u> |
| <u>L1</u> | cache with (buffer near4 (flush\$3 or clear\$3 or clean\$4 or empt\$4 or remov\$4 or eliminat\$3)) | 451   | <u>L1</u> |

END OF SEARCH HISTORY

**Set Name Query**

side by side

**Hit Count Set Name**

result set

*DB=USPT; PLUR=YES; OP=ADJ*

|            |   |      |            |
|------------|---|------|------------|
| <u>L19</u> | L18 and l14   | 23   | <u>L19</u> |
| <u>L18</u> | l13 same (buffer\$3 or queu\$4)   | 70   | <u>L18</u> |
| <u>L17</u> | L16 and l15   | 19   | <u>L17</u> |
| <u>L16</u> | L12 with tag with (modifi\$8 or updat\$4)   | 137  | <u>L16</u> |
| <u>L15</u> | L14 and l13   | 136  | <u>L15</u> |
| <u>L14</u> | L12 with (data near6 (modifi\$8 or updat\$4))   | 515  | <u>L14</u> |
| <u>L13</u> | L12 with (data near4 (cast\$4 or castout or purg\$4 or flush\$4 or clear\$4 or empt\$4 or remov\$4 or replac\$4 or eliminat\$4))                    | 341  | <u>L13</u> |
| <u>L12</u> | ((("L2" or second or secondary or external or (store adj2 in)) near4 cache)   | 7850 | <u>L12</u> |
| <u>L11</u> | l10 and (cache with (flush\$3 with (buffer\$3 or queu\$4)))   | 19   | <u>L11</u> |
| <u>L10</u> | l9 and (tag with cache with (modifi\$8 or updat\$4))  | 96   | <u>L10</u> |
| <u>L9</u>  | L8 and (tag near4 (memory or storage or RAM))   | 132  | <u>L9</u>  |
| <u>L8</u>  | L6 and (((("L2" or second or secondary or external or (store adj2 in)) near4 cache) near8 (modifi\$8 or updat\$4))                                  | 333  | <u>L8</u>  |
| <u>L7</u>  | L6 and (((("L2" or second or secondary or external) near4 cache) near8 (modifi\$8 or updat\$4))   | 332  | <u>L7</u>  |
| <u>L6</u>  | cache and ((buffer\$3 or queu\$4) near8 (cast\$4 or castout or purg\$4 or flush\$4 or clear\$4 or empt\$4 or remov\$4 or replac\$4 or eliminat\$4)) | 5533 | <u>L6</u>  |
| <u>L5</u>  | (09651488 or 5594876).pn.   | 1    | <u>L5</u>  |
| <u>L4</u>  | l3 and ((buffer\$3 or queu\$4) near8 (cast\$4 or castout or purg\$4 or flush\$4 or clear\$4 or empt\$4 or remov\$4 or replac\$4 or eliminat\$4))    | 14   | <u>L4</u>  |
| <u>L3</u>  | L2 same (cast\$4 or castout or purg\$4 or flush\$4 or clear\$4 or empt\$4 or remov\$4 or replac\$4 or eliminat\$4)                                  | 21   | <u>L3</u>  |
| <u>L2</u>  | (store adj2 in) with cache  | 40   | <u>L2</u>  |
| <u>L1</u>  | (store adj2 in) near4 cache   | 35   | <u>L1</u>  |

END OF SEARCH HISTORY

**Set Name Query**  
side by side

**Hit Count Set Name**  
result set

*DB=PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=ADJ*

|            |  |     |            |
|------------|--|-----|------------|
| <u>L15</u> | L14 and (cache with (updat\$4 or modifi\$8 or chang\$4 or alter\$4 or dirty))  | 7   | <u>L15</u> |
| <u>L14</u> | L13 and (buffer\$4 or queu\$4 or register or ((temporary or transition) near4 (storage or memory or area)))  | 15  | <u>L14</u> |
| <u>L13</u> | L12 with (cast\$4 or castout or purg\$4 or flush\$4 or clear\$4 or empt\$4 or remov\$4 or replac\$4 or eliminat\$4 or writeback or (write back) or (store back)) | 18  | <u>L13</u> |
| <u>L12</u> | (store-in or (store in) or (store adj in)) near6 cache   | 134 | <u>L12</u> |

*DB=USPT; PLUR=YES; OP=ADJ*

|            |   |      |            |
|------------|---|------|------------|
| <u>L11</u> | L9 and (buffer\$4 or queu\$4 or register or ((temporary or transition) near4 (storage or memory or area)))  | 43   | <u>L11</u> |
| <u>L10</u> | L9 and (tag with (updat\$4 or modifi\$8 or chang\$4 or alter\$4 or dirty))  | 11   | <u>L10</u> |
| <u>L9</u>  | L8 and (cache with (updat\$4 or modifi\$8 or chang\$4 or alter\$4 or dirty))  | 45   | <u>L9</u>  |
| <u>L8</u>  | L7 and miss   | 46   | <u>L8</u>  |
| <u>L7</u>  | L5 with (cast\$4 or castout or purg\$4 or flush\$4 or clear\$4 or empt\$4 or remov\$4 or replac\$4 or eliminat\$4 or writeback or (write back) or (store back)) | 56   | <u>L7</u>  |
| <u>L6</u>  | L5 with (cast\$4 or castout or purg\$4 or flush\$4 or clear\$4 or empt\$4 or remov\$4 or replac\$4 or eliminat\$4)  | 39   | <u>L6</u>  |
| <u>L5</u>  | (store-in or (store in) or (store adj in)) near6 cache  | 150  | <u>L5</u>  |
| <u>L4</u>  | L3 and ((host or processor or cpu) with L1 with (updat\$4 or modifi\$8 or chang\$4 or alter\$4))  | 56   | <u>L4</u>  |
| <u>L3</u>  | L2 and (cache with (((flush\$3 or castout or cast\$3 or purg\$3 or clear\$4 or posted) near8 (buffer\$3 or queu\$4)) or (temporary near4 (storage or memory)))) | 179  | <u>L3</u>  |
| <u>L2</u>  | L1 with (cast\$4 or castout or purg\$4 or flush\$4 or clear\$4 or empt\$4 or remov\$4 or replac\$4 or eliminat\$4)  | 950  | <u>L2</u>  |
| <u>L1</u>  | ("L2" or second or secondary or external or (store adj2 in)) near4 cache  | 7850 | <u>L1</u>  |

END OF SEARCH HISTORY


[> home](#) [> about](#) [> feedback](#) [> login](#)

US Patent &amp; Trademark Office



Try the *new* Portal design  
Give us your opinion after using it.

## Search Results

Search Results for: [(store-in or write-back or copyback or store back) <near/4> cache and (buffer or queue) <sentence> (cast\* or castout or purg\* or flush\* or clear\* or empt\*) <sentence> (modifi\* or updat\* or alter\* or chang\*)]

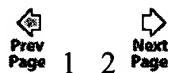
Found 23 of 122,783 searched.

## Search within Results


[> Advanced Search](#)
[> Search Help/Tips](#)

Sort by: [Title](#) [Publication](#) [Publication Date](#) [Score](#) [Binder](#)


Results 1 - 20 of 23 [short listing](#)



- 1 [Organization and performance of a two-level virtual-real cache hierarchy](#) 100%  
 W. H. Wang , J.-L. Baer , H. M. Levy  
**ACM SIGARCH Computer Architecture News , Proceedings of the 16th annual international symposium on Computer architecture April 1989**  
 Volume 17 Issue 3  
 We propose and analyze a two-level cache organization that provides high memory bandwidth. The first-level cache is accessed directly by virtual addresses. It is small, fast, and, without the burden of address translation, can easily be optimized to match the processor speed. The virtually-addressed cache is backed up by a large physically-addressed cache; this second-level cache provides a high hit ratio and greatly reduces memory traffic. We show how the second-level cache can be easily e ...
- 2 [Articles: Division of Labor in Embedded Systems](#) 100%  
 Ivan Goddard  
**Queue April 2003**  
 Volume 1 Issue 2
- 3 [Pipeline behavior prediction for superscalar processors by abstract interpretation](#) 100%  
 Jörn Schneider , Christian Ferdinand  
**ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1999 workshop on Languages, compilers, and tools for embedded systems May 1999**  
 Volume 34 Issue 7  
 For real time systems not only the logical function is important but also the timing behavior, e. g. hard real time systems must react inside their deadlines. To guarantee this it is necessary to know upper bounds for the worst case execution times (WCETs). The accuracy of the


prediction of WCETs depends strongly on the ability to model the features of the target processor. Cache memories, pipelines and parallel functional units are architectural components which are responsible for the speed gain ...

- 4** An effective write policy for software coherence schemes

 Y.-C. Chen , A. V. Veidenbaum

**Proceedings of the 1992 ACM/IEEE conference on Supercomputing** December 1992


100%
- 5** An improved replacement strategy for function caching

 William Pugh

**Proceedings of the 1988 ACM conference on LISP and functional programming** January 1988

Function caching is the technique of remembering previous function calls and avoiding the cost of recomputing them. Function caching provides a simple way of implementing dynamic programming algorithms and can provide a facility for incremental computation. Previous discussions of function caching have generally relied on the user to purge items from the function cache or have proposed a strategy such as least-recently-used without any analysis of the appropriateness of the ...


99%
- 6** Efficient and realistic simulation of disk cache performance

 John F. Cigas

**Proceedings of the 21st annual symposium on Simulation** January 1988

This paper describes an improved method for evaluating disk cache performance using trace driven simulation. This method differentiates between reads and writes in the trace data which results in higher miss ratios than when all traced events are treated alike. It allows for simulating various update policies and update intervals, physical blocks that are part of different files at different times, and the optimum replacement policy GOPT. These methods are applied to traces from a VAX 11/78 ...


99%
- 7** Session S4.1: power in memory and network processors: Embedded cache architecture with programmable write buffer support for power and performance flexibility

 Afzal Malik , Bill Moyer , Roger Zhou

**Proceedings of the international conference on Compilers, architecture, and synthesis for embedded systems** October 2002

Next generation portable devices are placing stringent requirements on overall system power and performance. Voice recognition, streaming video and high speed wireless internet access are just some of the features being incorporated in these handheld electronic gadgets. The M<sub>6</sub>CORE M341-S processor has been designed for high performance and cost sensitive portable products as well as for high end embedded control applications. M341-S obtains increased performance over the M<sub>6</sub>CORE M2 and M310 families ...

99%
- 8** The Wisconsin multicube: a new large-scale cache-coherent multiprocessor

 J. R. Goodman , P. J. Woest

**ACM SIGARCH Computer Architecture News , Proceedings of the 15th Annual International Symposium on Computer architecture** May 1988

Volume 16 Issue 2

The Wisconsin Multicube, is a large-scale, shared-memory multiprocessor architecture that employs a snooping cache protocol over a grid of buses. Each processor has a conventional

99%

(SRAM) cache optimized to minimize memory latency and a large (DRAM) snooping cache optimized to reduce bus traffic and to maintain consistency. The large snooping cache should guarantee that nearly all the traffic on the buses will be generated by I/O and accesses to shared data. The p ...

9 I/O reference behavior of production database workloads and the TPC benchmarks 99%

 analysis at the logical level

Windsor W. Hsu , Alan Jay Smith , Honesty C. Young

**ACM Transactions on Database Systems (TODS)** March 2001

Volume 26 Issue 1

As improvements in processor performance continue to far outpace improvements in storage performance, I/O is increasingly the bottleneck in computer systems, especially in large database systems that manage huge amounts of data. The key to achieving good I/O performance is to thoroughly understand its characteristics. In this article we present a comprehensive analysis of the logical I/O reference behavior of the peak production database workloads from ten of the world's largest corporatio ...

10 Journaling with ReisersFS

99%

 Chris Mason

**Linux Journal** February 2001

Mason gives a tour through the Reiser File System: its features and construction.

11 An architectural perspective on a memory access controller

99%

 M. Freeman

**Proceedings of the 14th annual international symposium on Computer architecture** June 1987

In this paper a CMOS memory access controller chip is described that provides the basis for achieving high-performance 68020-based (68030-based) systems. This controller matches the speed of the memory system to that of the microprocessor by providing a virtual cache mechanism where address translations are only required when there is a cache miss. This mechanism also facilitates the construction of shared-memory multiprocessor system where the controller manages ...

12 Prefetching in segmented disk cache for multi-disk systems

99%

 Valery Soloviev

**Proceedings of the fourth workshop on I/O in parallel and distributed systems: part of the federated computing research conference** May 1996

13 Avoiding conflict misses dynamically in large direct-mapped caches

99%

 Brian N. Bershad , Dennis Lee , Theodore H. Romer , J. Bradley Chen


**Proceedings of the sixth international conference on Architectural support for programming languages and operating systems** November 1994

Volume 29 , 28 Issue 11 , 5

This paper describes a method for improving the performance of a large direct-mapped cache by reducing the number of conflict misses. Our solution consists of two components: an inexpensive hardware device called a Cache Miss Lookaside (CML) buffer that detects conflicts by recording and summarizing a history of cache misses, and a software policy within the operating system's virtual memory system that removes conflicts by dynamically

remapping pages whenever large numbers of conflict miss ...

**14** A coherent distributed file cache with directory write-behind 99%


 Timothy Mann , Andrew Birrell , Andy Hisgen , Charles Jerian , Garret Swart

**ACM Transactions on Computer Systems (TOCS)** May 1994

Volume 12 Issue 2

Extensive caching is a key feature of the Echo distributed file system. Echo client machines maintain coherent caches of file and directory data and properties, with write-behind (delayed write-back) of all cached information. Echo specifies ordering constraints on this write-behind, enabling applications to store and maintain consistent data structures in the file system even when crashes or network faults prevent some writes from being completed. In this paper we describe ...

**15** High-speed buffering for variable length operands 99%


 H. L. Tredennick , T. A. Welch

**ACM SIGARCH Computer Architecture News , Proceedings of the 4th annual symposium on Computer architecture** March 1977

Volume 5 Issue 7

Variable word length processing is valuable for data base manipulations, editing functions in time-sharing systems, input-output data formatting, and vector operations, but current computer architectures seldom provide efficient means for manipulating variable length operands. A specialized computer architecture has been proposed to deal with the problems of variable length byte string processing. Operand buffering is a key part of the proposed architecture because the buffer: (1) replaces ...


**16** Architecture 2: An interleaved cache clustered VLIW processor 99%

 Enric Gibert , Jesús Sánchez , Antonio González

**Proceedings of the 16th international conference on Supercomputing** June 2002

Clustered microarchitectures are becoming a common organization due to their potential to reduce the penalties caused by wire delays and power consumption. Fully-distributed architectures are particularly effective to deal with these constraints, and besides they are very scalable. However, the distribution of the data cache memory poses a significant challenge and may be critical for performance. In this work, a distributed data cache VLIW architecture based on an interleaved cache organization ...

**17** The development of the MU5 computer system 99%


 R. N. Ibbett , P. C. Capon

**Communications of the ACM** January 1978

Volume 21 Issue 1

Following a brief outline of the background of the MU5 project, the aims and ideas for MU5 are discussed. A description is then given of the instruction set, which includes a number of features conducive to the production of efficient compiled code from high-level language source programs. The design of the processor is then traced from the initial ideas for an associatively addressed "store" to the final multistage pipeline structure involving a prediction mechanism for in ...

**18** Fetch directed instruction prefetching 99%


 Glenn Reinman , Brad Calder , Todd Austin

**Proceedings of the 32nd annual ACM/IEEE international symposium on  
Microarchitecture** November 1999

Instruction supply is a crucial component of processor performance. Instruction prefetching has been proposed as a mechanism to help reduce instruction cache misses, which in turn can help increase instruction supply to the processor. In this paper we examine a new instruction prefetch architecture called Fetch Directed Prefetching, and compare it to the performance of next-line prefetching and streaming buffers. This architecture uses a decoupled b ...

**19** Microarchitecture support for improving the performance of load target prediction

99%


 Chung-Ho Chen , Akida Wu

**Proceedings of the 30th annual ACM/IEEE international symposium on  
Microarchitecture** December 1997

Presents a load target prediction scheme that mitigates the impact of load latency for modern microprocessors. The scheme uses a cache-like buffer to provide the base address, offset and operand size at the instruction fetching stage of a pipeline so that a load target address can be computed earlier at the decode stage. With the dynamic use of a load stride, the scheme has achieved a prediction rate that is 15% higher than a previously proposed approach. By providing a 128-entry direct-mapped l ...

**20** Architectural primitives for a scalable shared memory multiprocessor



99%

 Joonwon Lee , Umakishore Ramachandran

**Proceedings of the third annual ACM symposium on Parallel algorithms and  
architectures** June 1991

---

Results 1 - 20 of 23    [short listing](#)

   
Prev Page 1 2 Next Page

---

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2003 ACM, Inc.

**IEEE Xplore®**  
RELEASE 1.5Welcome  
United States Patent and Trademark Office[Help](#) [FAQ](#) [Terms](#) [IEEE](#) [Quick Links](#)[» Search Results](#)

Welcome to IEEE Xplore®

Your search matched **[0]** of **[983096]** documents.

- ☐ Home
- ☐ What Can I Access?
- ☐ Log-out

## Tables of Contents

- ☐ Journals & Magazines
- ☐ Conference Proceedings
- ☐ Standards

## Search

- ☐ By Author
- ☐ Basic
- ☐ Advanced

## Member Services

- ☐ Join IEEE
- ☐ Establish IEEE Web Account
- ☐ Access the IEEE Member Digital Library

Print Format

You may refine your search by editing the current search expression or entering a new one the text box. Then click search Again.

(store-in or write-back or copyback or store back) <near/4> cache and (second or two) <

**OR**

Use your browser's back button to return to your original search page.

**Results:****No documents matched your query.**

[Home](#) | [Log-out](#) | [Journals](#) | [Conference Proceedings](#) | [Standards](#) | [Search by Author](#) | [Basic Search](#) | [Advanced Search](#)  
[Join IEEE](#) | [Web Account](#) | [New this week](#) | [OPAC Linking Information](#) | [Your Feedback](#) | [Technical Support](#) | [Email Alerting](#)  
[No Robots Please](#) | [Release Notes](#) | [IEEE Online Publications](#) | [Help](#) | [FAQ](#) | [Terms](#) | [Back to Top](#)

## Welcome to IEEE Xplore®

- ☐ Home
- ☐ What Can I Access?
- ☐ Log-out

## Tables of Contents

- ☐ Journals & Magazines
- ☐ Conference Proceedings
- ☐ Standards

## Search

- ☐ By Author
- ☐ Basic
- ☐ Advanced

## Member Services

- ☐ Join IEEE
- ☐ Establish IEEE Web Account
- ☐ Access the IEEE Member Digital Library

1) Enter a single keyword, phrase, or Boolean expression.  
Example: acoustic imaging (means the phrase acoustic imaging plus any stem variations)

2) Limit your search by using search operators and field codes, if desired.

Example: optical (fiber fibre) ti

3) Limit the results by selecting Search Options.

4) Click Search. See [Search Examples](#)

(store-in or write-back or copyback or store back) <near/4> cache and (second or two) <near/2> (buffer or queue) <sentence> (cast\$ or castout or purg\$ or flush\$ or clear\$ or empt\$)

Note: This function returns plural and suffixed forms of the keyword(s).

Search operators: [More](#)

Field codes: au (author), ti (title), ab (abstract), jn (publication name), de (index term) [More](#)

**Search Options:****Select publication types:**

- ☒ IEEE Journals
- ☒ IEE Journals
- ☒ IEEE Conference proceedings
- ☒ IEE Conference proceedings
- ☒ IEEE Standards

**Select years to search:**

From year:  to

**Organize search results by:**

Sort by:

In:  order

List  Results per page


[> home](#) [> about](#) [> feedback](#) [> login](#)

US Patent &amp; Trademark Office



Try the *new* Portal design  
Give us your opinion after using it.

## Search Results

Search Results for: [(store-in or write-back or copyback or store back) <near/4> cache and (second or two) <near/2> (buffer or queue) <sentence> (cast\* or castout or purg\* or flush\* or clear\* or empt\*)]

Found 17 of 122,783 searched.

Search within Results


[> Advanced Search](#)
[> Search Help/Tips](#)

Sort by: [Title](#) [Publication](#) [Publication Date](#) [Score](#) [Binder](#)

Results 1 - 17 of 17 [short listing](#)

1 [Multiprocessor Architectures For Concurrent Programs](#)

100%


 Per Brinch Hansen

**Proceedings of the 1978 annual conference** December 1978

This paper proposes a hierarchical multiprocessor architecture for real-time programs written in a concurrent programming language. The use of processes and monitors leads to a multiprocessor system in which each processor has a local store dedicated to a single process. The processors share a common store that contains the monitors. To avoid congestion in the common store the processes and monitors are partitioned into subsystems that share a hierarchy of common stores. The main goal is to ...

2 [Multiprocessor architectures for concurrent programs](#)

100%

 Per Brinch Hansen


**ACM SIGARCH Computer Architecture News** December 1978

Volume 7 Issue 4

This paper proposes a hierarchical multiprocessor architecture for real-time programs written in a concurrent programming language. The use of processes and monitors leads to a multiprocessor system in which each processor has a local store dedicated to a single process. The processors share a common store that contains the monitors. To avoid congestion in the common store the processes and monitors are partitioned into subsystems that share a hierarchy of common stores. The main goal is to deve ...

3 [Runtime identification of cache conflict misses: The adaptive miss buffer](#)

100%

 Jamison D. Collins , Dean M. Tullsen

**ACM Transactions on Computer Systems (TOCS)** November 2001


Volume 19 Issue 4

This paper describes the miss classification table, a simple mechanism that enables the processor or memory controller to identify each cache miss as either a conflict miss or a

capacity (non-conflict) miss. The miss classification table works by storing part of the tag of the most recently evicted line of a cache set. If the next miss to that cache set has a matching tag, it is identified as a conflict miss. This technique correctly identifies 88&percent; of misses. Several applications of this i ...

#### 4 Third Generation Computer Systems

100%

 Peter J. Denning


**ACM Computing Surveys (CSUR)** December 1971

Volume 3 Issue 4

The common features of third generation operating systems are surveyed from a general view, with emphasis on the common abstractions that constitute at least the basis for a &ldquo;theory&rdquo; of operating systems. Properties of specific systems are not discussed except where examples are useful. The technical aspects of issues and concepts are stressed, the nontechnical aspects mentioned only briefly. A perfunctory knowledge of third generation systems is presumed.

#### 5 Delay streams for graphics hardware

99%

 Timo Aila , Ville Miettinen , Petri Nordlund

**ACM Transactions on Graphics (TOG)** July 2003

Volume 22 Issue 3

In causal processes decisions do not depend on future data. Many well-known problems, such as occlusion culling, order-independent transparency and edge antialiasing cannot be properly solved using the traditional causal rendering architectures, because future data may change the interpretation of current events. We propose adding a *delay stream* between the vertex and pixel processing units. While a triangle resides in the delay stream, subsequent triangles generate occlusion information. ...

#### 6 Protocol verification using reachability analysis: the state space explosion problem and relief strategies

99%

 F. J. Lin , P. M. Chu , M. T. Liu

**ACM SIGCOMM Computer Communication Review , Proceedings of the ACM workshop on Frontiers in computer communications technology** August 1987

Volume 17 Issue 5

Reachability analysis has proved to be one of the most effective methods in verifying correctness of communication protocols based on the state transition model. Consequently, many protocol verification tools have been built based on the method of reachability analysis. Nevertheless, it is also well known that state space explosion is the most severe limitation to the applicability of this method. Although researchers in the field have proposed various strategies to relieve this intricate p ...

#### 7 Detailed design and evaluation of redundant multithreading alternatives

99%

 Shubhendu S. Mukherjee , Michael Kontz , Steven K. Reinhardt

**ACM SIGARCH Computer Architecture News** May 2002

Volume 30 Issue 2

Exponential growth in the number of on-chip transistors, coupled with reductions in voltage levels, makes each generation of microprocessors increasingly vulnerable to transient faults. In a multithreaded environment, we can detect these faults by running two copies of the same program as separate threads, feeding them identical inputs, and comparing their outputs, a

technique we call Redundant Multithreading (RMT). This paper studies RMT techniques in the context of both single- and dual-process ...

8 Predictor-directed stream buffers

99%

Timothy Sherwood , Suleyman Sair , Brad Calder

**Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture** December 2000

9 Performance of database workloads on shared-memory systems with out-of-order processors

99%

Parthasarathy Ranganathan , Kourosh Gharachorloo , Sarita V. Adve , Luiz André Barroso

**Proceedings of the eighth international conference on Architectural support for programming languages and operating systems** October 1998

Volume 33 , 32 Issue 11 , 5

Database applications such as online transaction processing (OLTP) and decision support systems (DSS) constitute the largest and fastest-growing segment of the market for multiprocessor servers. However, most current system designs have been optimized to perform well on scientific and engineering workloads. Given the radically different behavior of database workloads (especially OLTP), it is important to re-evaluate key system design decisions in the context of this important class of applicatio ...

10 Trace-driven memory simulation: a survey

99%

Richard A. Uhlig , Trevor N. Mudge

**ACM Computing Surveys (CSUR)** June 1997

Volume 29 Issue 2

As the gap between processor and memory speeds continues to widen, methods for evaluating memory system designs before they are implemented in hardware are becoming increasingly important. One such method, trace-driven memory simulation, has been the subject of intense interest among researchers and has, as a result, enjoyed rapid development and substantial improvements during the past decade. This article surveys and analyzes these developments by establishing criteria for evaluating trac ...

11 Classification and performance evaluation of instruction buffering techniques

99%

Lizyamma Kurian , Paul T. Hulina , Lee D. Coraor , Dhamir N. Mannai

**ACM SIGARCH Computer Architecture News , Proceedings of the 18th annual international symposium on Computer architecture** April 1991

Volume 19 Issue 3

12 The NYU Ultracomputer—designing a MIMD, shared-memory parallel machine

99%

(Extended Abstract)

Allan Gottlieb , Ralph Grishman , Clyde P. Kruskal , Kevin P. McAuliffe , Larry Rudolph , Marc Snir

**Proceedings of the 9th annual symposium on Computer Architecture** April 1982

We present the design for the NYU Ultracomputer, a shared-memory MIMD parallel machine composed of thousands of autonomous processing elements. This machine uses an enhanced message switching network with the geometry of an Omega-network to approximate the ideal behavior of Schwartz's paracomputer model of computation and to implement efficiently the important fetch-and-add synchronization primitive. We outline the hardware that would be required to build a 4096 processor system using 1990' ...

- 13 The holodeck ray cache: an interactive rendering system for global illumination in nondiffuse environments 99%  
Gregory Ward , Maryann Simmons  
**ACM Transactions on Graphics (TOG)** October 1999  
Volume 18 Issue 4  
We present a new method for rendering complex environments using interactive, progressive, view-independent, parallel ray tracing. A four-dimensional holodeck data structure serves as a rendering target and caching mechanism for interactive walk-throughs of nondiffuse environments with full global illumination. Ray sample density varies locally according to need, and on-demand ray computation is supported in a parallel implementation. The holodeck file is stored on disk and ...
- 14 Multiple instruction issue in the NonStop cyclone processor 99%  
Robert W. Horst , Richard L. Harris , Robert L. Jardine  
**ACM SIGARCH Computer Architecture News , Proceedings of the 17th annual international symposium on Computer Architecture** May 1990  
Volume 18 Issue 3  
This paper describes the architecture for issuing multiple instructions per clock in the NonStop Cyclone Processor. Pairs of instructions are fetched and decoded by a dual two-stage prefetch pipeline and passed to a dual six-stage pipeline for execution. Dynamic branch prediction is used to reduce branch penalties. A unique microcode routine for each pair is stored in the large duplexed control store. The microcode controls parallel data paths optimized for executing the most frequent instr ...
- 15 The NYU ultracomputer&mdash;designing a MIMD, shared-memory parallel machine 99%  
Allan Gottlieb , Ralph Grishman , Clyde P. Kruskal , Kevin P. McAuliffe , Larry Rudolph , Marc Snir  
**25 years of the international symposia on Computer architecture (selected papers)**  
August 1998
- 16 ECOSystem: managing energy as a first class operating system resource 98%  
Heng Zeng , Carla S. Ellis , Alvin R. Lebeck , Amin Vahdat  
**Tenth international conference on architectural support for programming languages and operating systems on Proceedings of the 10th international conference on architectural support for programming languages and operating systems (ASPLOS-X)**  
October 2002  
Volume 37 , 30 , 36 Issue 10 , 5 , 5  
Energy consumption has recently been widely recognized as a major challenge of computer systems design. This paper explores how to support energy as a first-class operating system resource. Energy, because of its global system nature, presents challenges beyond those of conventional resource management. To meet these challenges we propose the Currentcy Model that unifies energy accounting over diverse hardware components and enables fair allocation of available energy among applications. Our par ...
- 17 The design and implementation of a progressive on-demand image dissemination system for very large images 98%  
Michael J. Owen , Andrew K. Lui , Edward H. S. Lo , Mark W. Grigg

**Australian Computer Science Communications , Proceedings of the 24th Australasian conference on Computer science January 2001**

Volume 23 Issue 1

The use of progressive, on-demand image dissemination techniques can support efficient dissemination of very large images across networks. In this paper we examine the effectiveness of various design options in developing such on-demand dissemination systems. We show that the choice of the design options can have a profound impact on the efficient use of client, server, and network resources. Based on our performance evaluation experiments, we recommend that efficient dissemination can be achiev ...

---

Results 1 - 17 of 17    [short listing](#)

---

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2003 ACM, Inc.

**IEEE Xplore®**  
RELEASE 1.5Welcome  
United States Patent and Trademark Office[Help](#) [FAQ](#) [Terms](#)  
[IEEE Peer Review](#)[Quick Links](#)[» Advanced Search](#)

## Welcome to IEEE Xplore®

- ☐ Home
- ☐ What Can I Access?
- ☐ Log-out

## Tables of Contents

- ☐ Journals & Magazines
- ☐ Conference Proceedings
- ☐ Standards

## Search

- ☐ By Author
- ☐ Basic
- ☐ Advanced

## Member Services

- ☐ Join IEEE
- ☐ Establish IEEE Web Account
- ☐ Access the IEEE Member Digital Library

1) Enter a single keyword, phrase, or Boolean expression.  
Example: acoustic imaging (means the phrase acoustic imaging plus any stem variations)

2) Limit your search by using search operators and field codes, if desired.

Example: optical (fiber fibre) ti

3) Limit the results by selecting Search Options.

4) Click Search. See [Search Examples](#)

(store-in or write-back or copyback or store back) <near/4> cache and (buffer or queue) <sentence> (cast\$ or castout or purg\$ or flush\$ or clear\$ or empt\$) <sentence> (modifi\$ or updat\$ or alter\$ or

Note: This function returns plural and suffixed forms of the keyword(s).

Search operators: [More](#)

Field codes: au (author), ti (title), ab (abstract), jn (publication name), de (index term) [More](#)

**Search Options:****Select publication types:**

- ☒ IEEE Journals
- ☒ IEE Journals
- ☒ IEEE Conference proceedings
- ☒ IEE Conference proceedings
- ☒ IEEE Standards

**Select years to search:**

From year:  to

**Organize search results by:**

Sort by:

In:  order

List  Results per page

## Welcome to IEEE Xplore®

- ☐ Home
- ☐ What Can I Access?
- ☐ Log-out

## Tables of Contents

- ☐ Journals & Magazines
- ☐ Conference Proceedings
- ☐ Standards

## Search

- ☐ By Author
- ☐ Basic
- ☐ Advanced

## Member Services

- ☐ Join IEEE
- ☐ Establish IEEE Web Account
- ☐ Access the IEEE Member Digital Library

 Print FormatYour search matched **[0]** of **[983096]** documents.

You may refine your search by editing the current search expression or entering a new one the text box. Then click search Again.

(store-in or write-back or copyback or store back) <near/4> cache and (buffer or queue)

[Search Again](#)**OR**

Use your browser's back button to return to your original search page.

**Results:****No documents matched your query.**

[Home](#) | [Log-out](#) | [Journals](#) | [Conference Proceedings](#) | [Standards](#) | [Search by Author](#) | [Basic Search](#) | [Advanced Search](#)  
[Join IEEE](#) | [Web Account](#) | [New this week](#) | [OPAC Linking Information](#) | [Your Feedback](#) | [Technical Support](#) | [Email Alerting](#)  
[No Robots Please](#) | [Release Notes](#) | [IEEE Online Publications](#) | [Help](#) | [FAQ](#) | [Terms](#) | [Back to Top](#)